**PHARMACY MANAGEMENT SYSTEM**

**A PROJECT REPORT**

**for**

**Mini Project (KCA353)**

**Session (2023-24)**

**Submitted by**

**Prachi Sharma**

**(University Roll No: 2200290140110)**

Submitted in partial fulfilment of the

Requirements for the Degree of

**MASTER OF COMPUTER APPLICATION**

Under the Supervision of

Dr. Shashank Bhardwaj

Associate Professor



**Submitted to**

**Department Of Computer Applications**

KIET Group of Institutions, Ghaziabad

Uttar Pradesh-201206

**(January** 2024**)**

**CERTIFICATE**

Certified that **Prachi Sharma 2200290140110** has/ have carried out the project work having “**Pharmacy management system**” (**Mini Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under Dr Shashank Bhardwaj supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date:**

**Prachi Sharma (2200290140110)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

**Dr Shashank Bhardwaj Dr. Arun Tripathi**

**Associate Professor Head**

**Department of Computer Applications Department of Computer Applications**

**KIET Group of Institutions, Ghaziabad KIET Group of Institutions, Ghaziabad**

**Pharmacy Management System**

**Prachi Sharma**

**ABSTRACT**

The Pharmacy Management System (PMS) is a comprehensive and user-friendly web-based application designed to revolutionize the operations of pharmacies. Traditional pharmacy processes often involve manual and time-consuming tasks, leading to inefficiencies, errors, and challenges in maintaining accurate inventory records. The Pharmacy Management System addresses these issues by automating key aspects of pharmacy operations, including inventory management, prescription processing, and sales tracking.

The primary objective of the Pharmacy Management System is to enhance the overall efficiency and customer service within pharmacy settings. By leveraging modern web technologies and a robust database backend, the system aims to streamline workflow processes, minimize errors in prescription handling, and provide valuable insights through detailed sales reports and analytics.

**ACKNOWLEDGEMENTS**

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, Dr Shashank Bhardwaj for his guidance, help, and encouragement throughout my project work.

Their enlightening ideas, comments, and suggestions, Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Head Department of Computer Applications, for his insightful comments and administrative help on various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness .

**Prachi Sharma**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
|  | Certificate | i |
|  | Abstract | ii |
|  | Acknowledgements | iii |
|  | Table of Contents | iv |
|  | List of Abbreviations | ix |
|  | List of Tables | xii |
|  | List of Figures | xiii |
| 1 | **INTRODUCTION** | 1-16 |
|  | **1.1 BACKGROUND** | 2 |
|  | **1.2 OBJECTIVES OF THE PROJECT** | 4 |
|  | 1.2.1 Automation | 4 |
|  | 1.2.2 Inventory control | 7 |
|  | **1.3 SCOPE OF THE PROJECT** | 12 |
|  | 1.3.1 Drug Inventory Management | 13 |
|  | 1.3.2 Sales and Billing Functionality | 14 |
|  | 1.3.3 User Authentication and Access Control | 15 |
|  | **1.4 SIGNIFICANCE OF THE PROJECT** | 16 |
|  | 1.4.1 Project Overview | 17-36 |
|  | **1.5 HARDWARE & SOFTWARE SPECIFICATION** | 18 |
|  | 1.5.1 Hardware Specification | 18 |
|  | 1.5.2 Software Specification | 35 |
| 2 | **LITERATURE REVIEW** | 35 |
|  | **2.1 OVERVIEW** | 36 |
|  | **2.2 EVOLUTION OF PHARMACY MANAGEMENT SYSTEMS** |  |
|  | **2.3 KEY FEATURES AND FUNCTIONALITIES** |  |
|  | 2.3.1 Smart Inventory Control |  |
|  | 2.3.4 Secure User Authentication |  |
|  | 2.3.5 User-Friendly Interface |  |
|  | **2.4 TECHNOLOGICAL FRAMEWORKS** |  |
|  | 2.4.1 PHP (Hypertext Preprocessor) |  |
|  | 2.4.2 XAMPP (Cross-Platform, Apache, MariaDB, PHP, and Perl) |  |
|  | **2.5 CHALLENGES AND SOLUTIONS IN PHARMACY MANAGEMENT** |  |
|  | 2.5.1 Challenges |  |

|  |  |  |
| --- | --- | --- |
|  | 2.5.2 Solutions |  |
|  | **FEASIBILITY STUDY** |  |
|  | **3.1 PROJECT DESCRIPTION** |  |
|  | **3.2 MARKET ANALYSIS** |  |
|  | **3.3 TECHNICAL FEASIBILITY** |  |
|  | **3.4 OPERATIONAL FEASIBILITY** |  |
|  | 3.4.1 Financial Feasibility |  |
|  | 3.4.2 Legal and Regulatory Compliance |  |
|  | 3.4.3 Risk Analysis |  |
|  | 3.4.4 Timeline and Implementation Plan |  |
| 4 | REQUIREMENT ANALYSIS |  |
|  | 4.1 FUNCTIONAL REQUIREMENTS |  |
|  | 4.1.1 Medication Management |  |
|  | 4.1.2 Patient Management |  |
|  | 4.1.3 Reporting and Analytics |  |
|  | 4.2 NON-FUNCTIONAL REQUIREMENTS |  |
|  | 4.2.1 Performance |  |
|  | 4.2.2 Reliability |  |
|  | 4.2.3 Security |  |
|  | 4.2.4 Scalability |  |
|  | **4.3 USE CASE SCENARIOS** |  |
|  | 4.3.1 User Authentication and Authorization |  |
|  | 4.3.2 Inventory Management |  |
|  | 4.3.3. Prescription Processing |  |
|  | 4.3.4 Sales and Billing |  |
|  | 4.3.5 Patient Records |  |
|  | 4.4 DATA REQUIREMENTS |  |
|  | 4.4.1 Patient Data |  |
|  | 4.4.2 Medication Data |  |
|  | 4.4.3 Sales and Billing Data |  |
|  | 4.4.4 Backup and Recovery Data |  |
| 5 | SYSTEM DESIGN |  |
|  | 5.1 HIGH LEVEL SYSTEM ARCHITECHTURE |  |
|  | 5.2 DATABASE DESIGN |  |
|  | 5.3 CLASS DIAGRAM OR ENTITY RELATIONSHIP DIAGRAM |  |
|  | 5.3.1 Class Diagram |  |
|  | 5.3.2 Entity relationship diagram(ERD) |  |
|  | 5.3.3 Data flow diagram |  |
|  |  |  |
| 6 | DESIGN |  |
|  | 6.1 LOGIN PAGE |  |
|  | 6.2 DASHBOARD |  |
|  | 6.3 ADD MANUFACTURER |  |
|  | 6.4 MANAGE MANUFACTURER |  |
|  | 6.5 ADD CATEGORIES |  |
|  | 6.6 MANAGE CATEGORIES |  |
|  | 6.8 INVOICES |  |
|  | 6.8.1 Add Invoices |  |
|  | 6.8.2 Manage invoices |  |
|  | 6.9 Report |  |
| 7 | IMPLEMENTATION |  |
|  | 7.1 Overview of PHP |  |
|  | 7.1.1 Key Features |  |
|  | 7.2 DATABASE SETUP AND CONNECTIVITY |  |
|  | 7.3 SECURITY MEASURES IMPLEMENTED |  |
|  | 7.3.1 Database Security |  |
| 8 | TESTING |  |
|  | 8.1 TESTING METHODOLOGIES |  |
|  | 8.1.1 Unit Testing |  |
|  | 8.1.2 Integration Testing |  |
|  | 8.1.3 System Testing |  |
|  | 8.1.4 Acceptance Testing |  |
|  | 8.1.5 Security Testing |  |
|  | 8.2 Test cases and results |  |
|  | 8.3 Bug tracking and resolution |  |
|  | 8.3.1 Bug Identification |  |
|  | 8.3.2 Bug Logging |  |
|  | 8.3.3 Bug Resolution |  |
|  | 8.3.4 Testing Fixes |  |
|  | 8.3.5 Verification and Validation |  |
|  | Conclusion |  |
|  | References & Bibliography |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Name of Figure** | **Page No.** |
| 3.1 | Use Case Diagram | 11 |
| 3.2 | Entity-Relationship Diagram | 12 |
| 3.3 | Level – 0 DFD | 15 |
| 3.4 | Level – 1 DFD | 16 |
| 3.5 | Level – 2 DFD for Subject Update | 16 |
| 3.6 | Level – 2 DFD for Question Update | 17 |
| 3.7 | Level – 2 DFD for Exam Process | 17 |
| 6.1 | Home Page | 23 |
| 6.2 | Student Home Page | 23 |
| 6.3 | Student Sign-Up Page | 24 |
| 6.4 | Student Login Page | 24 |
| 6.5 | Student Dashboard | 25 |
| 6.6 | Student Examination Page | 25 |
| 6.7 | Student Marks Page | 26 |
| 6.8 | Admin Login Page | 26 |
| 6.9 | Admin Dashboard | 27 |
| 6.10 | Admin Student Section | 27 |
| 6.11 | Admin Course Section | 28 |
| 6.12 | Admin Add Course Section | 28 |
| 6.13 | Admin View Course Section | 29 |
| 6.14 | Admin Question Section | 29 |
| 6.15 | Admin Add Question Section | 30 |
| 6.16 | Admin Available Question Section | 30 |
| 6.17 | Admin View Question Section | 31 |

**CHAPTER 1**

**INTRODUCTION**

Welcome to the Pharmacy Management System project, an innovative solution poised to transform the landscape of pharmacy operations. In an era of advancing healthcare services, the efficient management of pharmaceutical processes is critical to meet the demands of an ever-evolving industry. This project harnesses the capabilities of PHP to deliver a sophisticated, user-friendly platform, addressing the complex needs of modern pharmacies. In a world where the intersection of healthcare and technology is increasingly integral, the meticulous management of pharmaceutical processes stands as a cornerstone for ensuring the well-being of communities. This project, driven by the dynamic capabilities of PHP, aspires to introduce a sophisticated, user-friendly platform designed to address the multifaceted challenges faced by modern pharmacies.

**1.1 BACKGROUND**

Pharmacies serve as crucial hubs in healthcare, ensuring the timely and accurate distribution of essential medications to the public. As the healthcare landscape becomes more intricate, the necessity for a streamlined and automated approach to manage pharmacy activities becomes increasingly apparent.Pharmacies, as vital pillars of healthcare provision, shoulder the responsibility of ensuring the timely and accurate distribution of life-saving medications. The complexity of this mission has grown exponentially, driven by advancements in medical science, increased demand for specialized pharmaceuticals, and the ever-expanding scope of healthcare services. This project emerges as a response to the imperative need for a streamlined, automated approach to navigate this intricate web of pharmaceutical activities.

**1.2 OBJECTIVES OF THE PROJECT**

This project sets forth with a clear set of objectives, each meticulously crafted to address the contemporary challenges faced by pharmacies:

**1.2.1 Automation**

The primary goal is to automate and simplify the intricate web of pharmaceutical data management, reducing reliance on manual processes and enhancing overall efficiency. A focus on accurate and efficient inventory management ensures that pharmacies can maintain optimal stock levels, preventing both shortages and excesses.

**1.2.2 Prescription processing**

Seamlessly processing prescriptions is at the core of the project, aiming to enhance the speed and accuracy of order fulfilment. The integration of technology into various facets of pharmacy operations is poised to enhance overall efficiency, offering a modernized approach to an age-old profession.

**1.3 SCOPE OF THE PROJECT**

Encompassing a wide array of functionalities, the scope of this project extends to cover critical aspects of pharmacy management:

**1.3.1 Drug Inventory Management**

The system will provide tools for efficient tracking and control of pharmaceutical stocks, ensuring pharmacies are well-equipped to meet the demands of their clientele. Streamlining the process of prescription processing, the system seeks to reduce processing times, minimize errors, and enhance overall prescription management.

**1.3.2 Sales and Billing Functionality**

The incorporation of a robust system for sales transactions and billing addresses the financial aspect of pharmacy operations, ensuring transparency and accuracy.

**1.3.3 User Authentication and Access Control**

Prioritizing security, the system will implement user authentication mechanisms and access controls to safeguard sensitive pharmaceutical data.

**1.4 SIGNIFICANCE OF THE PROJECT**

* By deploying an automated system, pharmacies can significantly improve their operational efficiency, reducing the time and effort spent on manual processes.
* Automation inherently minimizes the scope for human errors, especially in critical processes such as prescription processing and inventory management.
* A centralized platform for managing pharmaceutical data ensures that pharmacies can adhere to regulatory requirements, maintain accurate records, and respond swiftly to changing conditions.
* In the pursuit of these objectives, this Pharmacy Management System project emerges as not just a technological innovation but as a catalyst for enhancing the quality of healthcare services and contributing to the resilience and adaptability of pharmacies in an ever-evolving healthcare landscape.

**1.4.1 Project Overview**

* Front End            :     PHP (5.5)
* Back End             :     MYSQL

**1.5 HARDWARE & SOFTWARE SPECIFICATION**

**1.5.1 Hardware Specification**

* Processor            :     Intel Core Duo 2.0 GHz or higher.
* RAM                    :     Minimum1 GB or Greater.
* Hard disk            :    20 GB (Free Space).

**1.5.2 Software Specification**

* Software                      :    XAMPP.
* Operation System     :    Windows 7 or higher.

**CHAPTER 2**

**LITERATURE REVIEW**

**2.1 OVERVIEW**

In the pursuit of developing an effective Pharmacy Management System, a comprehensive exploration of existing literature is indispensable. This chapter endeavors to provide a thorough review of relevant studies, research papers, and technological frameworks that form the foundational knowledge for the proposed system. By examining the evolution of Pharmacy Management Systems, key features and functionalities, technological frameworks, challenges, solutions, case studies, and identifying gaps in the existing literature, this chapter aims to synthesize a holistic understanding of the current landscape and inform the design and development of the proposed system.

The project is all about creating a special tool, a Pharmacy Management System. Its job is to help pharmacies run smoothly and efficiently. We want to use technology to make things better for everyone involved.

We're using a language called PHP, like a magic spell for the internet, and databases, which are like super-organized shelves for storing pharmacy information. These tools will help our system be smart and organized.

**2.2 EVOLUTION OF PHARMACY MANAGEMENT SYSTEMS**

The historical progression of Pharmacy Management Systems has witnessed a paradigm shift from manual, paper-based processes to sophisticated, technology-driven solutions. Early pharmacy operations heavily relied on manual record-keeping, leading to inefficiencies, data inaccuracies, and increased susceptibility to errors. As technological advancements burgeoned, the healthcare industry recognized the need for automated systems to streamline processes, improve accuracy, and enhance overall efficiency. The evolution of Pharmacy Management Systems reflects a continual quest for innovation, driven by the growing complexities of pharmaceutical operations and the imperative to meet evolving healthcare standards.

Pharmacies did everything by hand, like keeping track of medicines and prescriptions using paper and pens. Then, smart folks had the idea to use computers, giving birth to Pharmacy Management Systems. These systems evolved to become super-smart assistants for pharmacies, ensuring efficient stock management and quick prescription processing. Now, our Pharmacy Management System project is taking this evolution further by using modern tools like PHP and databases to make the system even more efficient and powerful

**2.3 KEY FEATURES AND FUNCTIONALITIES**

**2.3.1 Smart Inventory Control**

The system meticulously tracks and manages the inventory of medicines. Ensures real-time updates on stock levels, preventing shortages and overstock situations. Accelerates the processing of prescriptions to minimize waiting times for customers and enhance the overall speed and accuracy of prescription fulfilment. Efficient manages sales transactions seamlessly, ensuring accurate and transparent billing and Keeps a comprehensive record of financial transactions for streamlined accounting.

**2.3.2 Secure User Authentication**

A robust user authentication system to control access to sensitive information .Ensures that only authorized personnel can interact with critical pharmacy data.

**2.3.3 User-Friendly Interface**

Features an intuitive and user-friendly interface for easy navigation. Enhances user experience, allowing pharmacy staff to interact with the system effortlessly.Adopts a forward-looking approach by seamlessly integrating with other healthcare systems.

**2.4 TECHNOLOGICAL FRAMEWORKS**

**2.4.1 PHP (Hypertext Pre-processor)**

PHP is a server-side scripting language widely used for web development. It allows the creation of dynamic and interactive web pages by embedding code within HTML. Role in the Project: Primary language for backend development in our Pharmacy Management System .Facilitates server-side processing, ensuring seamless interaction with the database.

**2.4.2 XAMPP (Cross-Platform, Apache, MariaDB, PHP, and Perl)**

XAMPP is an open-source software package providing a local server environment. It includes Apache (web server), MariaDB (database system), PHP, and Perl. Role in the Project: Serves as the local server for development, testing, and deployment. Ensures a comprehensive environment for PHP scripts to run, interact with the database, and serve web pages.

**2.5 CHALLENGES AND SOLUTIONS IN PHARMACY MANAGEMENT**

The implementation of Pharmacy Management Systems is not without its challenges. Common issues include concerns related to data security, discrepancies in inventory management, and the need for seamless integration with other healthcare systems.

**2.5.1 Challenges**

* Maintaining accurate inventory levels can be challenging, resulting in stockouts or overstock situations.
* Manual prescription processing can lead to delays in medication dispensing.
* Ensuring seamless integration with other healthcare systems may pose a challenge.

**2.5.2 Solutions**

* Implementation of the Pharmacy Management System automates data handling, minimizing errors and streamlining processes**.**
* The system includes a robust inventory control feature, providing real-time tracking and automated alerts for restocking. Implementation of user authentication and access controls within the system ensures data security and confidentiality

**CHAPTER 3**

**FEASIBILITY STUDY**

A feasibility study is a crucial step in the early stages of any project, including the development of a Pharmacy Management System using PHP. It helps assess the viability of the project and determines whether it is worth pursuing. Here's a general outline of what a feasibility study for a Pharmacy Management System might include:

**3.1 PROJECT DESCRIPTION**

In this section, provide a comprehensive and detailed overview of the Pharmacy Management System project. Elaborate on the specific goals and objectives, emphasizing how the proposed system aims to enhance the efficiency and effectiveness of pharmacy operations. Define the scope of the project by outlining the key features and functionalities expected, including but not limited to prescription management, inventory control, patient information tracking, billing, and reporting capabilities. Emphasize the potential positive impacts on workflow, customer service, and overall pharmacy management.

**3.2 MARKET ANALYSIS**

Conduct an in-depth analysis of the pharmaceutical industry, considering current market trends, demands, and the prevailing competitive landscape. Highlight the increasing reliance on digital solutions within the industry and showcase how a Pharmacy Management System aligns with these trends. Provide a thorough examination of existing systems in the market, assessing their strengths and weaknesses, and emphasize the unique selling points that the proposed system could offer.

**3.3 TECHNICAL FEASIBILITY**

Explore the technical intricacies of implementing the Pharmacy Management System. Discuss the necessary hardware, software, and networking infrastructure required for seamless operation. Evaluate the compatibility of the proposed system with existing technologies within the pharmacy and its ability to integrate with other healthcare systems, such as electronic health records (EHRs) or health information exchange (HIE) platforms.

**3.4 OPERATIONAL FEASIBILITY**

Delve into the operational aspects, analysing how well the Pharmacy Management System aligns with the current processes and workflows of the pharmacy. Identify potential bottlenecks, disruptions, or resistance from pharmacy staff during the implementation phase. Propose strategies to minimize any negative operational impacts and emphasize how the system can optimize day-to-day tasks.

**3.4.1 Financial Feasibility**

Provide a detailed breakdown of the financial aspects associated with the implementation of the Pharmacy Management System. This should include development costs, hardware and software expenses, training costs, and ongoing maintenance expenses. Conduct a thorough cost-benefit analysis, emphasizing the potential return on investment (ROI) and showcasing the long-term financial advantages of adopting the proposed system.

**3.4.2 Legal and Regulatory Compliance**

Scrutinize the legal and regulatory landscape, ensuring that the proposed Pharmacy Management System complies with all relevant laws and standards pertaining to healthcare data privacy and security. Highlight the system's ability to maintain patient confidentiality and comply with regulations such as Health Insurance Portability and Accountability Act (HIPAA) in the United States or other applicable regulations in different regions.

**3.4.3 Risk Analysis**

Undertake a comprehensive risk analysis, identifying potential challenges and obstacles that may arise during the project lifecycle. This could include resistance from pharmacy staff, technical challenges, or changes in regulatory requirements. Develop robust risk mitigation strategies and contingency plans to address these potential obstacles, demonstrating a proactive approach to risk management.

**3.4.4 Timeline and Implementation Plan**

Provide a detailed timeline outlining the key milestones and phases of the Pharmacy Management System project. Clearly define the development, testing, and implementation phases, ensuring that the timeline is realistic and achievable. Integrate key performance indicators (KPIs) to monitor progress and identify any deviations from the proposed schedule.

**CHAPTER 4**

**REQUIREMENT ANALYSIS**

**4.1 FUNCTIONAL REQUIREMENTS**

Developing a robust pharmacy management system requires a clear understanding of its functional requirements. These requirements define what the system should do to effectively support pharmacy operations and meet the needs of users, including pharmacists, nurses, doctors, and patients.

Here's a breakdown of key functional requirements for a pharmacy management system:

**4.1.1 Medication Management**:

• Inventory tracking: Maintain accurate records of medication quantities, locations, expiry dates, and batch numbers.

•Drug interaction checking: Integrate with drug databases to identify and alert potential medication interactions.

* + 1. **Patient Management:**

Patient profiles: Create and maintain patient profiles with demographic information, medical history, allergies, and medication lists.

Medication adherence monitoring: Track patient medication adherence and provide interventions when needed.

**4.1.3 Reporting and Analytics:**

* Generate reports on medication usage, inventory levels, dispensing trends, and other key metrics.
* Track and analyse pharmacy performance to ensure compliance with regulations and best practices.

**4.2 NON-FUNCTIONAL REQUIREMENTS:**

**4.2.1 Performance:**

The system should respond to user requests within 2 seconds to ensure a seamless user experience. It should support a minimum of 100 concurrent users during peak hours.

**4.2.2 Reliability:**

The system should have a 99.9% uptime, ensuring that it is available for use by pharmacy staff at all times. It should have a backup and recovery mechanism to prevent data loss in case of system failures. The system should be compatible with commonly used web browsers (e.g., Chrome, Firefox, Safari) to ensure widespread accessibility. It should be compatible with standard operating systems (e.g., Windows, macOS, Linux).

**4.2.3 Security:**

User data should be stored securely, and access to sensitive information should be restricted based on user roles. The system should encrypt communication between the server and clients.

**4.2.4 Scalability:**

The system should be designed to handle an increase in the number of pharmacies using the system. It should be scalable to accommodate a growing database of patients and medications. The user interface should be intuitive, and pharmacists should be able to perform common tasks with minimal training. The system should be accessible to users with disabilities, complying with accessibility standards.

**4.3 USE CASE SCENARIOS:**

**4.3.1 User Authentication and Authorization:**

**Use Case**: Pharmacist Login

**Scenario:** A pharmacist logs into the system with their credentials.

**Steps:**

Pharmacist enters username and password.

System verifies credentials.

If valid, the pharmacist gains access to the system.

**4.3.2 Inventory Management:**

**Use Case:** Update Medication Stock

**Scenario:** A pharmacist updates the stock of a medication.

**Steps:**

Pharmacist selects the medication to update.

Pharmacist enters the new quantity and updates the expiration date.

System updates the inventory database.

**4.3.3. Prescription Processing:**

**Use Case:** Process New Prescription

**Scenario:** A pharmacist processes a new prescription.

**Steps:**

Pharmacist enters prescription details (patient information, prescribed medications).

System checks for drug interactions.

If no issues, the system updates the patient's prescription history.

**4.3.4 Sales and Billing:**

**Use Case:** Generate Invoice

**Scenario:** A pharmacist generates an invoice for a customer.

**Steps:**

Pharmacist selects the medications sold.

System calculates the total cost.

Pharmacist generates and prints the invoice for the customer.

**4.3.5 Patient Records:**

**Use Case:** Search Patient Records

**Scenario:** A pharmacist searches for a patient's record.

**Steps:**

Pharmacist enters patient details (name, ID, etc.).

System retrieves and displays the patient's record.

These use case scenarios provide a high-level overview of how users (pharmacists in this case) interact with the Pharmacy Management System to perform various tasks. Each scenario outlines the steps involved in achieving a specific goal within the system, helping to understand the flow of interactions and functionalities.

**4.4 DATA REQUIREMENTS**

Data requirements for a Pharmacy Management System (PMS) involve specifying the types of data that the system needs to store, manage, and manipulate. Here are key data requirements for a PMS:

**4.4.1 Patient Data:**

Fields: Name, date of birth, contact information, address, insurance details.

Purpose: To maintain a comprehensive record of patients for prescription tracking and personalized care.

**4.4.2 Medication Data:**

Fields: Medication name, dosage, expiration date, manufacturer, stock quantity.

Purpose: To track and manage the pharmacy's inventory, ensuring accurate dispensing and timely reordering.

**4.4.3 Sales and Billing Data:**

Fields: Invoice ID, date of sale, medications sold, quantity, total cost, payment details.

Purpose: To track sales, generate invoices, and manage billing information for accounting purposes.

**User Data:**

Fields: User ID, username, password, role (pharmacist, administrator, manager), contact details.

Purpose: To manage user authentication, authorization, and track user activities within the system.

**Supplier Data:**

Fields: Supplier name, contact information, supplied medications, contract details.

Purpose: To manage relationships with suppliers, track medication sources, and facilitate reordering.

Fields: Medication pairs, interaction type, severity, recommended actions.

Purpose: To support the prescription processing functionality by identifying potential drug interactions.

**4.4.4 Backup and Recovery Data:**

Fields: Backup timestamps, backup location.

Purpose: To support data recovery in case of system failures or data loss.

It's important to define data structures, relationships, and constraints for these data elements to ensure data integrity, consistency, and security within the Pharmacy Management System. Additionally, considerations should be made for data privacy and compliance with relevant regulations.

**CHAPTER 5**

**SYSTEM DESIGN**

**5.1 HIGH-LEVEL SYSTEM ARCHITECTURE:**

The high-level system architecture for a Pharmacy Management System (PMS) typically involves multiple components working together to deliver the required functionalities. Here's a high-level overview:

* **User Interface (UI):**

Description: The front-end component that interacts with users (pharmacists, administrators, etc.).

Key Features:

User authentication and login.

Intuitive interfaces for prescription processing, inventory management, and other functionalities.

Dashboard for quick access to relevant information.

* **Application Layer:**

Description: The logic and business rules governing the system's functionalities.

Key Features:

Prescription processing algorithms.

Inventory management logic.

Billing and invoicing processes.

User authentication and authorization.

* **Database Management System (DBMS):**

Description: The backend database where all the system data is stored.

Key Features:

Structured storage of patient records, medication data, prescription details, etc.

Ensuring data integrity and consistency.

Support for efficient data retrieval.

* **Server:**

Description: The central server that hosts the application logic and manages communication with the database.

Key Features:

Handles user requests and processes business logic.

Manages data transactions with the database.

Ensures the system's overall performance and reliability.

* **Integration with External Systems:**

Description: Interfaces with other systems, such as healthcare databases, insurance systems, or external APIs.

Key Features:

Integration for checking drug interactions against external databases.

Data exchange for insurance verification.

Connection to suppliers for medication restocking.

* **Security Layer:**

Description: Ensures the security of the system, user data, and transactions.

Key Features:

User authentication mechanisms.

Encryption for secure data transmission.

Access control and authorization.

* **Reporting and Analytics:**

Description: Provides tools for generating reports and analyzing system data.

Key Features:

Reporting tools for administrators and managers.

Analytics for inventory management and sales performance.

* **Backup and Recovery**:

Description: Manages data backup and recovery processes to prevent data loss.

Key Features:

Scheduled backups of the database.

Recovery mechanisms in case of system failures.

User Management:

Description: Handles user roles, permissions, and authentication.

Key Features:

User account creation, modification, and deletion.

Role-based access control.

Communication Protocols:

This high-level architecture provides a foundation for designing and developing a Pharmacy Management System. It's important to consider scalability, maintainability, and security throughout the design process. The specific technologies and frameworks used will depend on the development team's expertise and the organization's requirements.

* 1. **DATABASE DESIGN**

**Entity-Relationship Diagram (ERD)**

Provide an ERD to visually represent the relationships between different entities in the database. This should include entities such as:

Patient

Medication

Prescription

Doctor

Pharmacist

Inventory

Supplier

Include relationships, cardinalities, and attributes for each entity.

**Database Tables**

**Patient Table:**

PatientID (Primary Key)

FirstName

LastName

DateOfBirth

ContactNumber

Address

**Medication Table:**

MedicationID (Primary Key)

MedicationName

Manufacturer

Dosage

PricePerUnit

**Prescription Table:**

PrescriptionID (Primary Key)

PatientID (Foreign Key)

DoctorID (Foreign Key)

DatePrescribed

Status (Pending, Filled, etc.)

**Doctor Table:**

DoctorID (Primary Key)

FirstName

LastName

Specialization

ContactNumber

**Pharmacist Table:**

PharmacistID (Primary Key)

FirstName

LastName

ContactNumber

**Supplier Table:**

SupplierID (Primary Key)

SupplierName

ContactNumber

Address

**Data Types and Constraints:**

Define the data types for each attribute and specify any constraints such as NOT NULL, UNIQUE, DEFAULT, etc.

**Indexing and Optimization:**

Discuss indexing strategies for critical fields to optimize query performance.

**Database Security:**

Outline the security measures implemented to protect sensitive information, including user authentication and authorization.

**Backup and Recovery:**

Describe the backup and recovery procedures to ensure data integrity and availability.

**Data Migration:**

Discuss how existing data, if any, will be migrated to the new system.

**Database Maintenance Plan:**

Provide a plan for routine maintenance tasks such as updates, patches, and performance monitoring.

**Database Testing:**

Outline the testing procedures for the database design, including unit testing, integration testing, and performance testing.

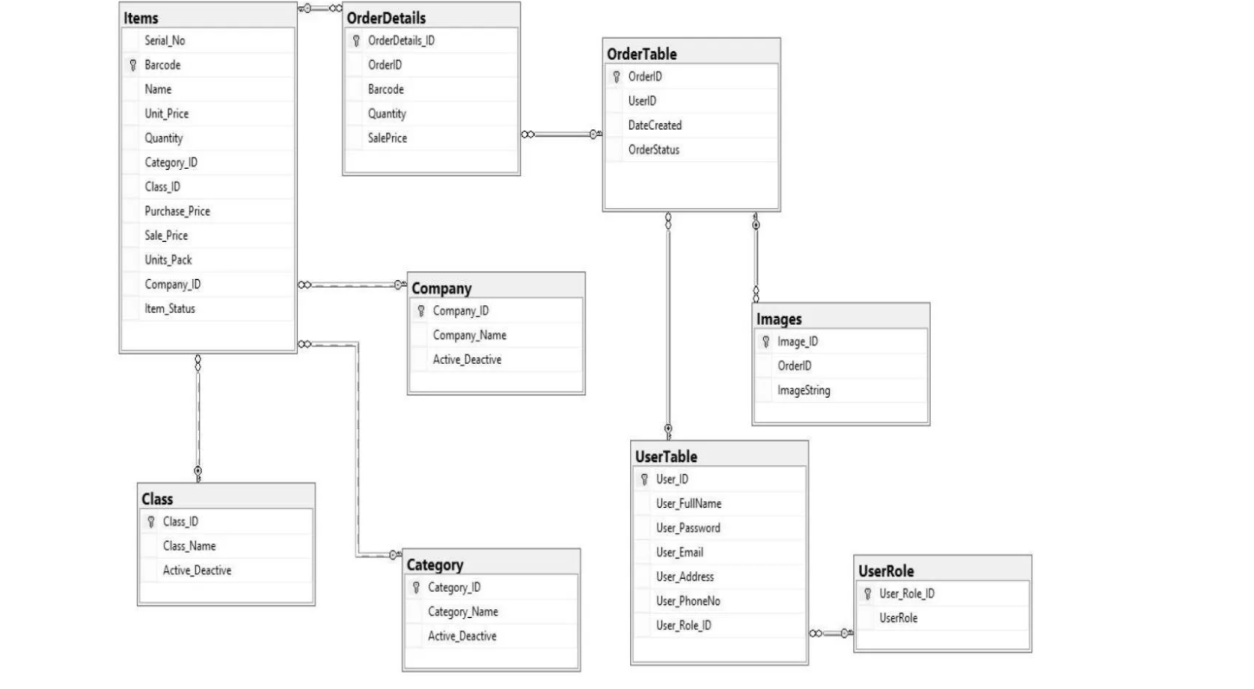
**5.3CLASS DIAGRAMS OR ENTITY-RELATIONSHIP DIAGRAMS**

**5.3.1 Class Diagrams**

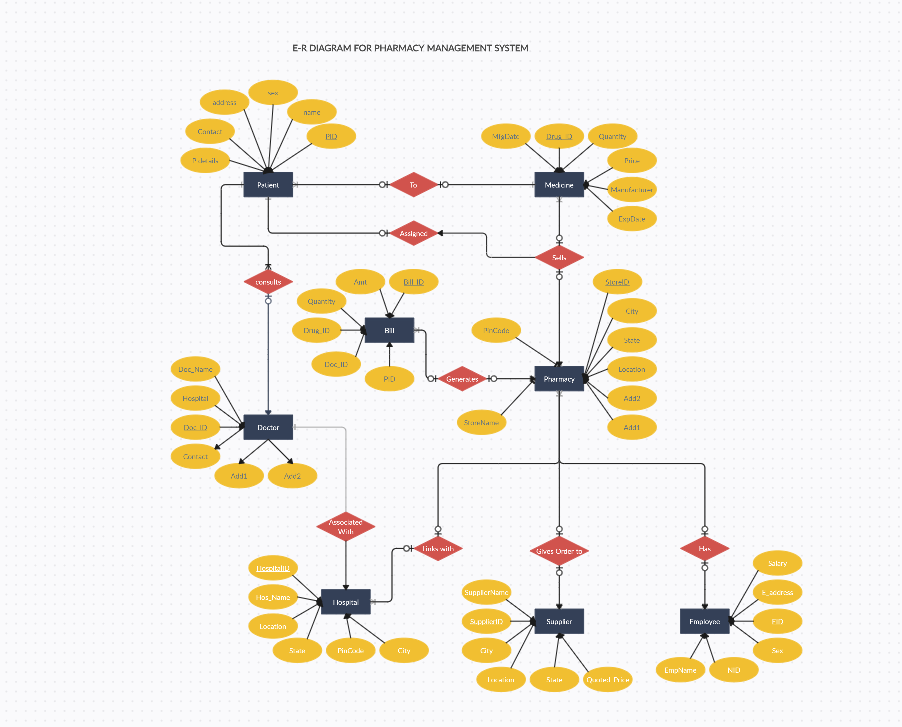
Class diagrams in the context of a pharmacy management system help to represent the static structure of the system by illustrating the classes, their attributes, methods, and relationships.

Key Elements:-

* **Classes:** Represent entities or objects in the system. Examples include Patient, Medication, Prescription, Doctor, Pharmacist, Inventory, and Supplier.
* **Attributes:** Properties of each class. For instance, the Patient class might have attributes like PatientID, FirstName, LastName, DateOfBirth, etc.
* **Methods/Operations:** Actions that can be performed on the classes. For example, the Prescription class might have methods like fillPrescription().
* **Relationships:** Connections between classes, indicating associations. For instance, a Prescription class might have associations with Patient and Doctor classes.



**5.3.2 Entity-Relationship Diagrams (ERD**

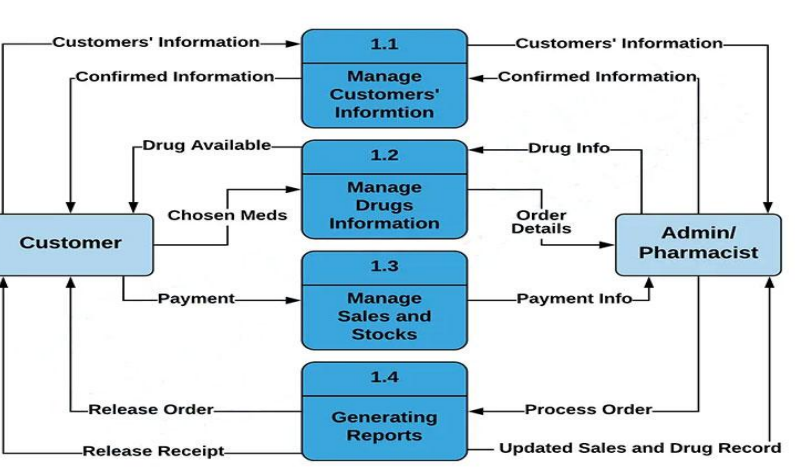
****

ERD focuses on depicting the relationships between entities within the system. It emphasizes the connections between different entities and their attributes.

Key Elements:

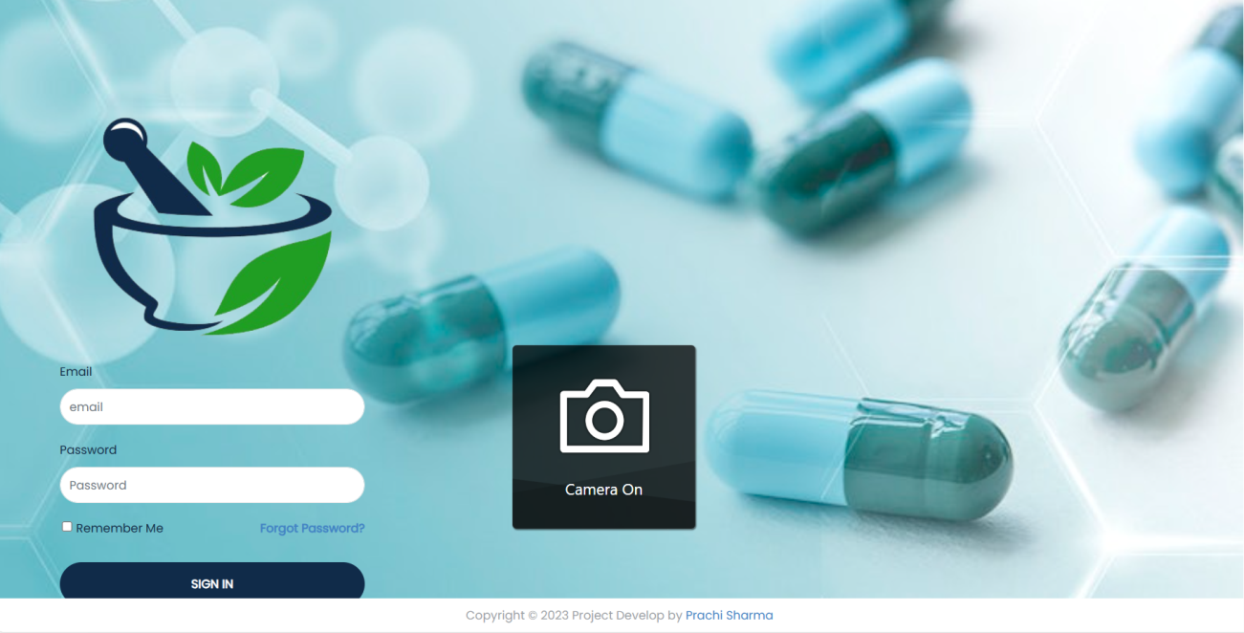
* **Entities:** Correspond to tables in the database. Patient, Medication, Prescription, Doctor, Pharmacist, Inventory, and Supplier would be entities in a pharmacy management system.
* **Attributes:** Characteristics of each entity. For instance, Patient entity might have attributes like PatientID, FirstName, LastName, etc.
* **Relationships:** Illustrate how entities are related to each other. For example, a Prescription entity might have relationships with Patient and Doctor entities.

**5.3.3 Data flow Diagram:**

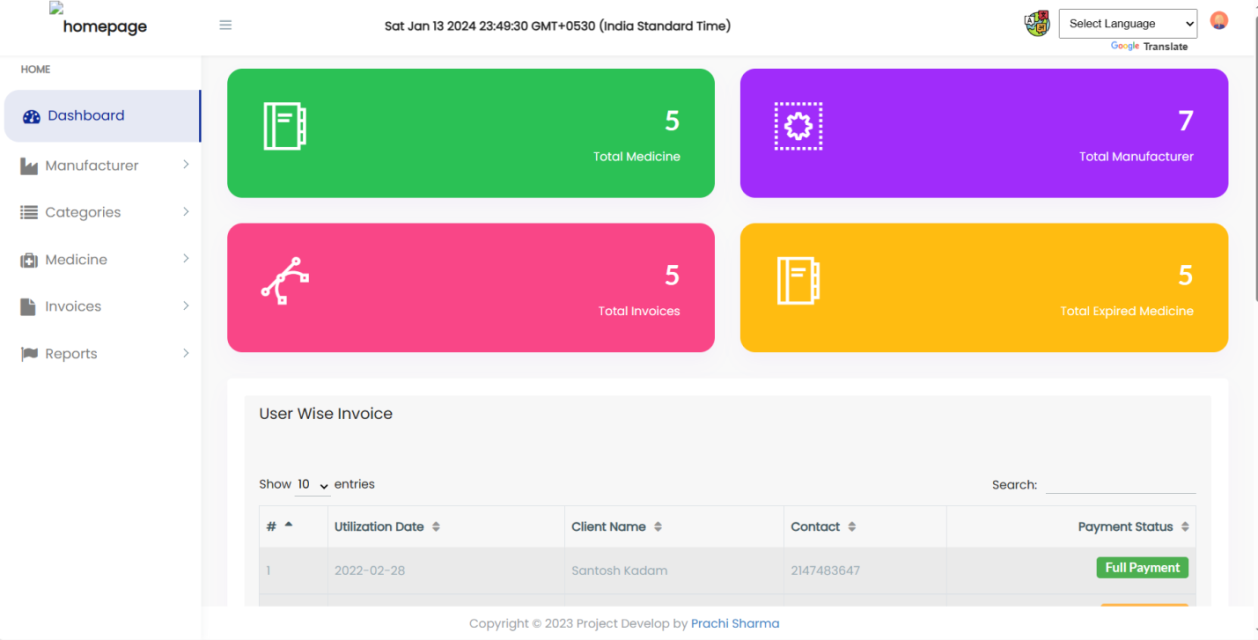
****

**CHAPTER 6**

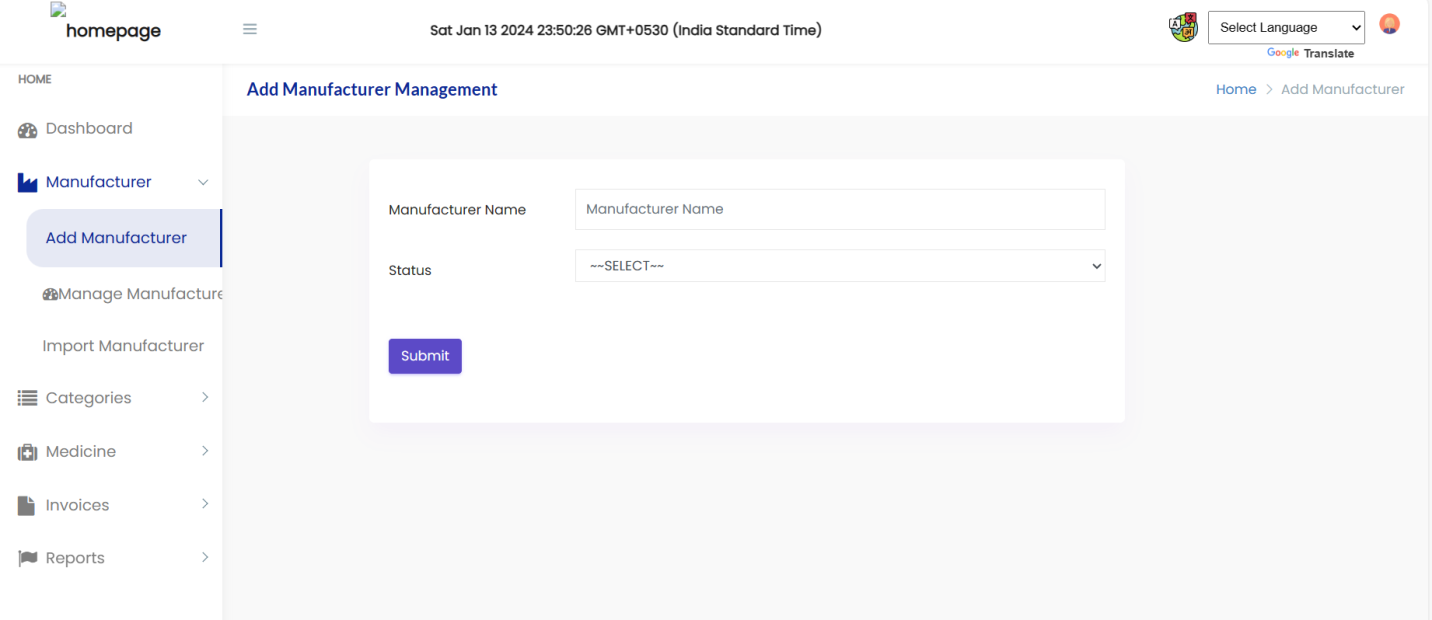
**DESIGN**

****

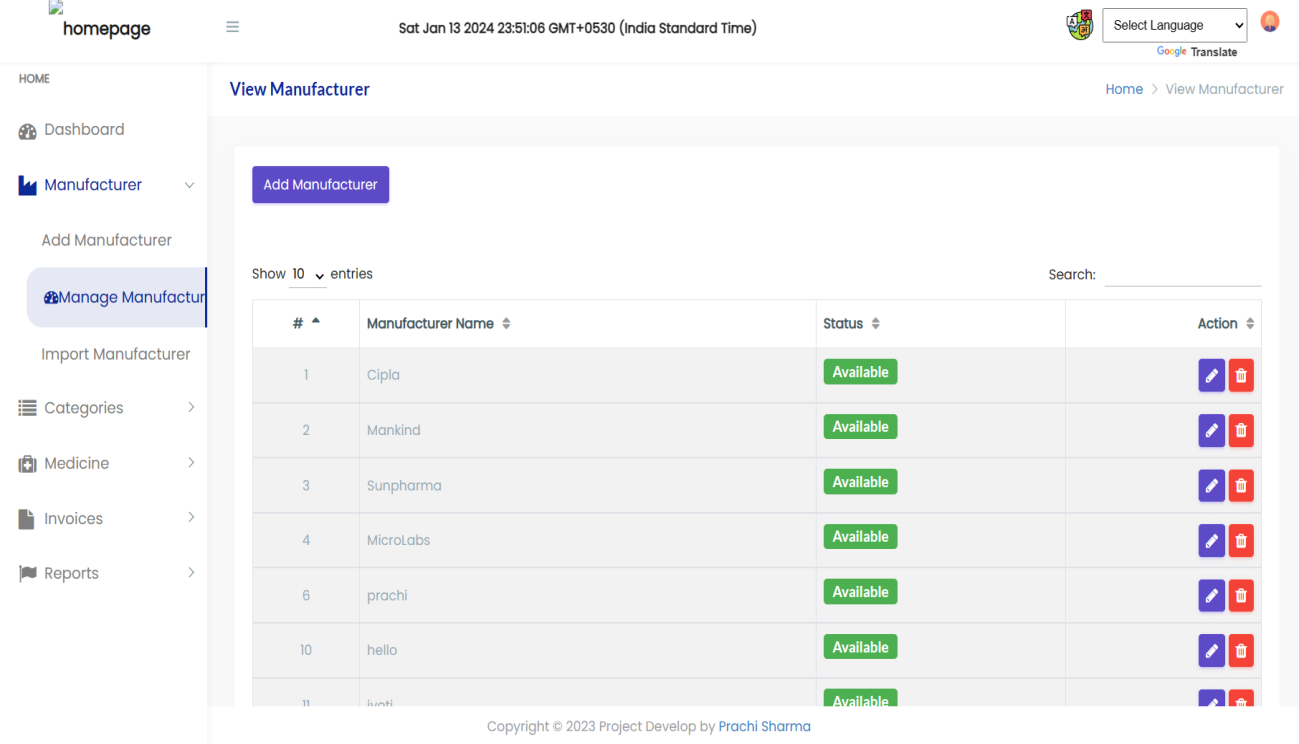
**Fig 6.1: LOGIN PAGE**

****

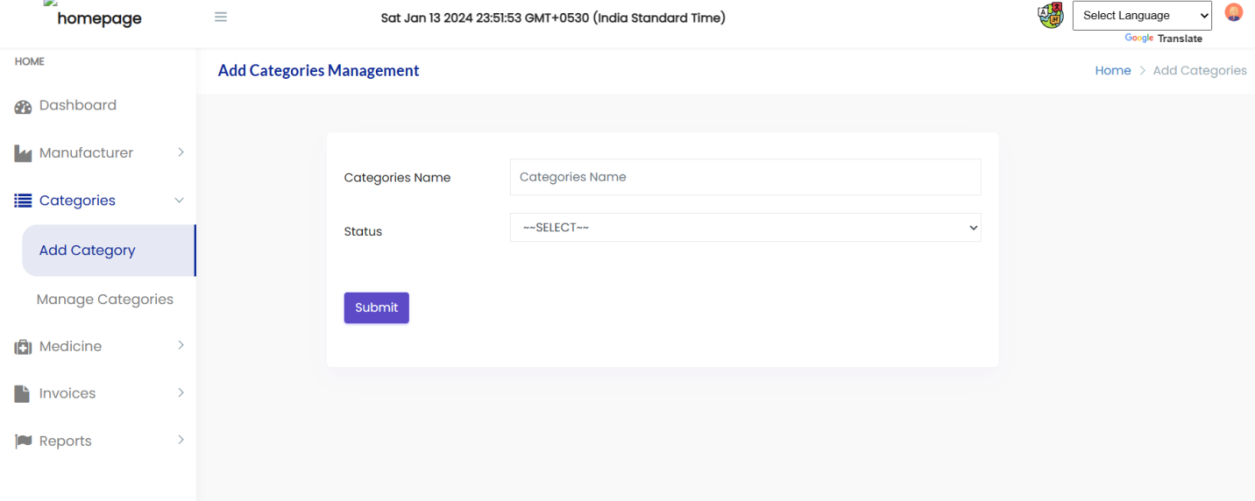
**Fig 6.2: DASHBOARD**



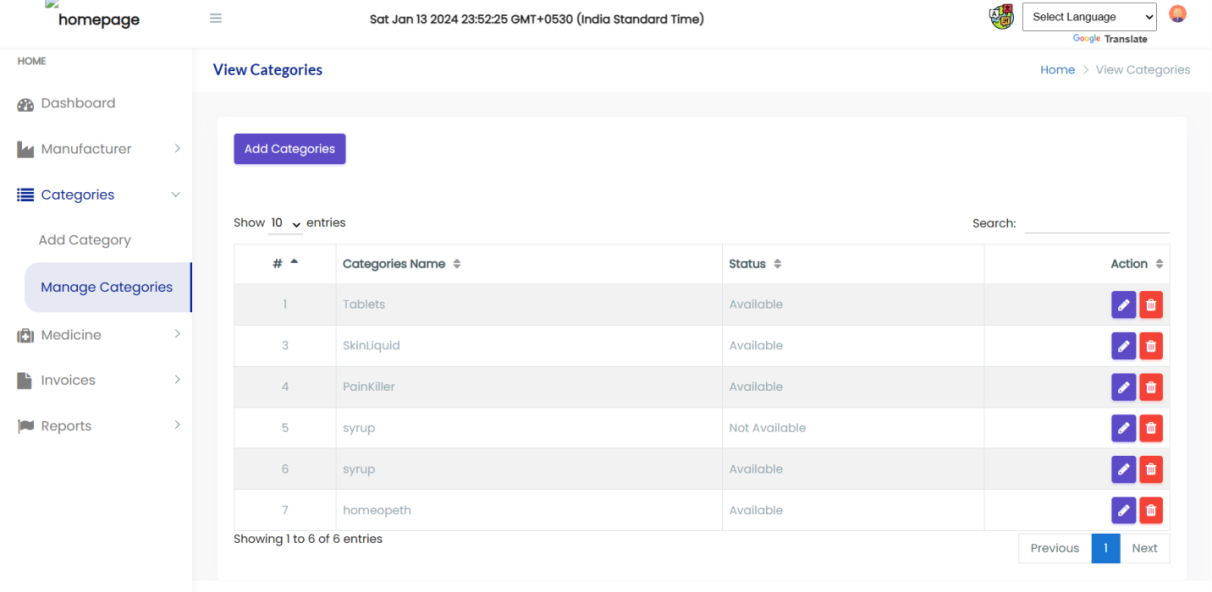
**Fig 6.3:ADD MANUFACTURER**

****

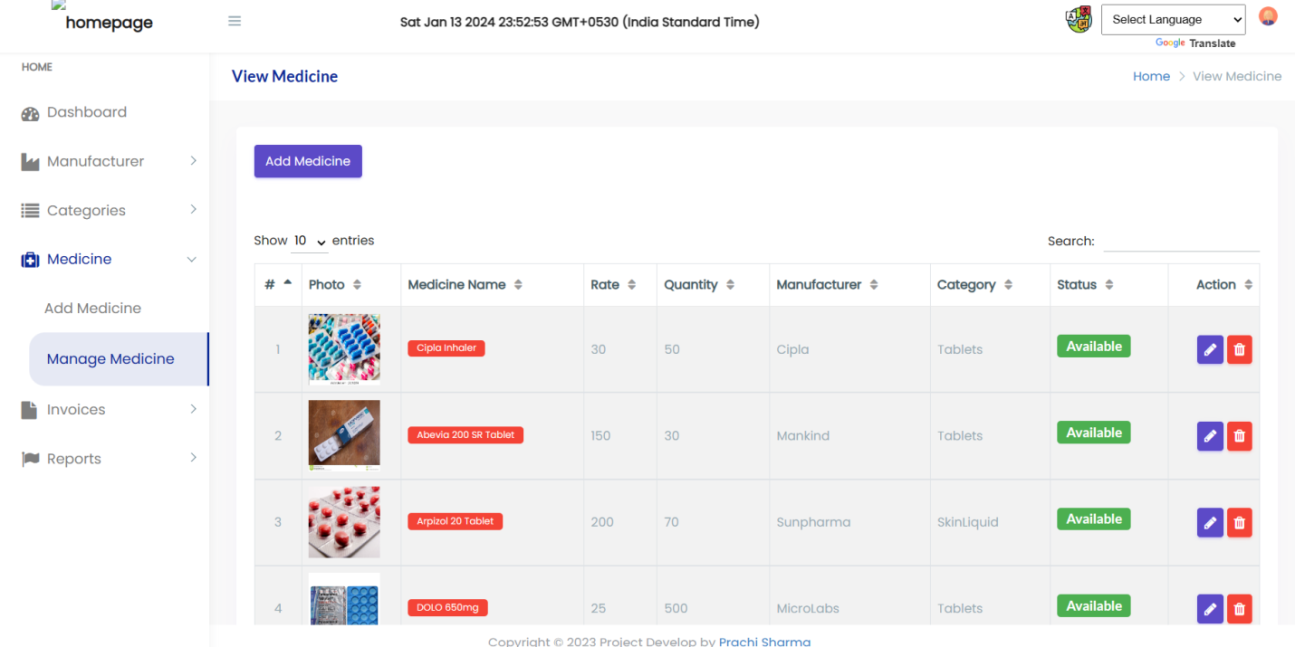
**Fig 6.4: MANAGE MANUFACTURER**

****

**Fig 6.5: ADD CATEGORIES**

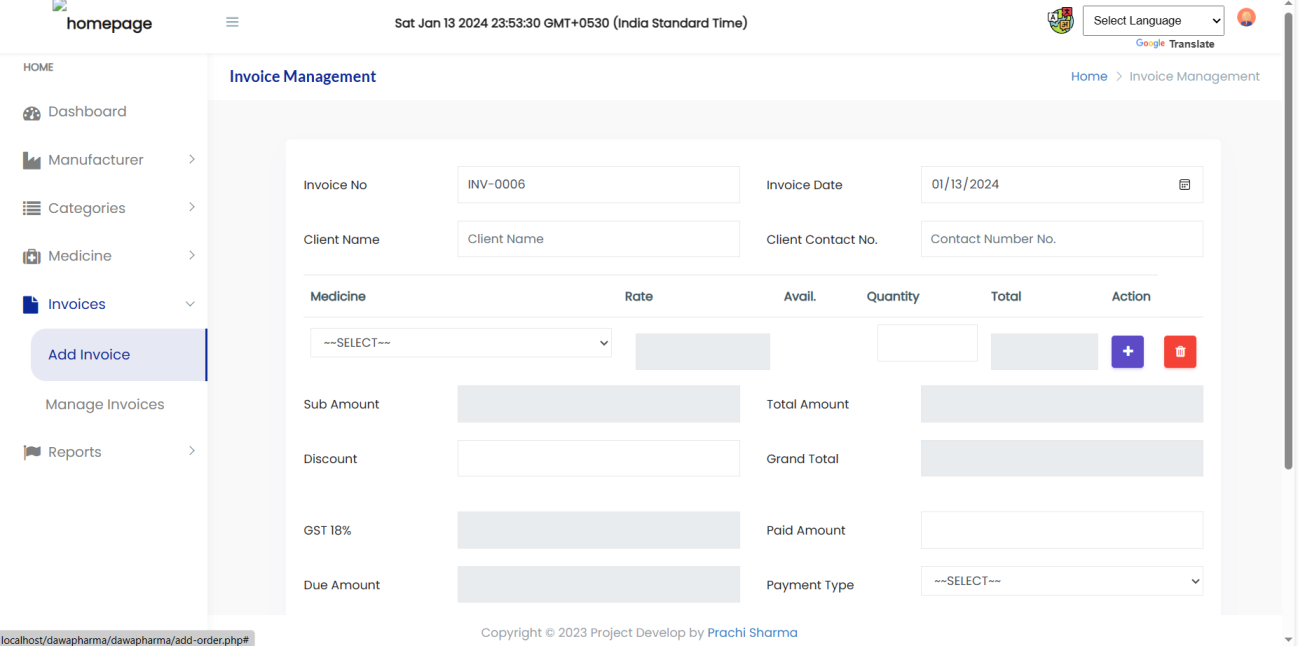
****

**Fig 6.6: MANAGE CATEGORIES**

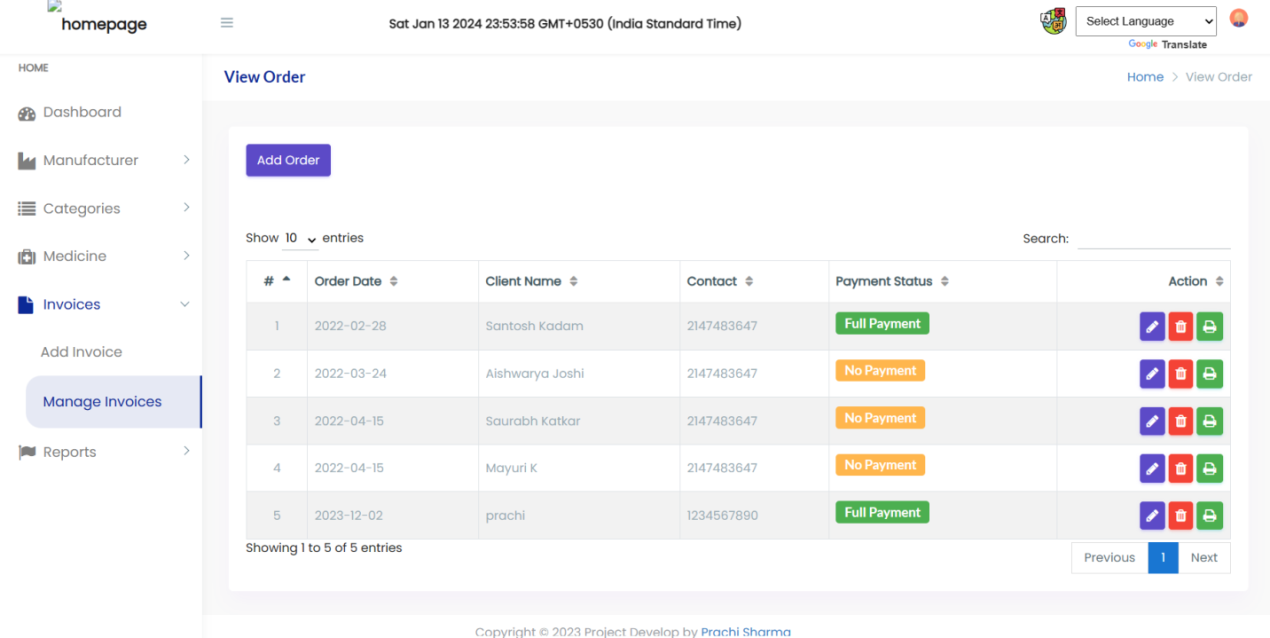
****

**Fig 6.7: MEDICINES**

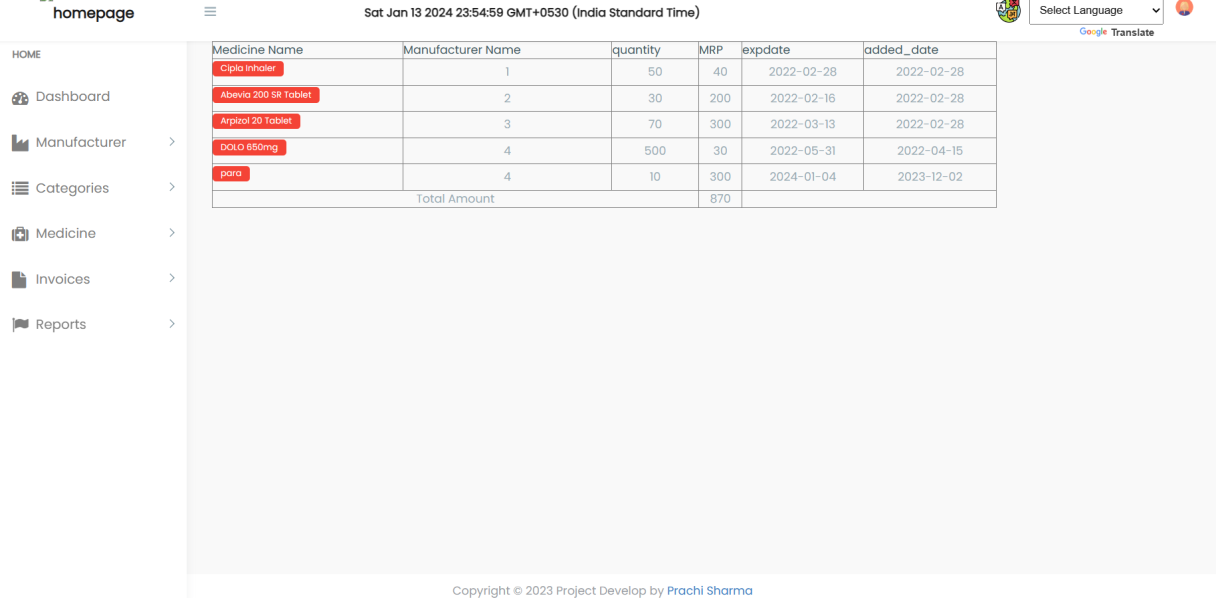
**6.8 INVOICES:**

****

**Fig 6.8.1: Add Invoices**

****

**Fig 6.8.2: Manage invoices**

****

**Fig 6.9: Report**

**CHAPTER 7**

**IMPLEMENTATION**

**7.1 Overview of PHP**

PHP (Hypertext Pre-processor) is a widely used, server-side scripting language designed for web development. It is particularly well-suited for creating dynamic web pages and applications. PHP code is embedded within HTML, allowing developers to seamlessly integrate dynamic content into static web pages.

**7.1.1 Key Features**

* **Open Source**: PHP is an open-source language, making it freely available for developers to use, modify, and distribute.
* **Server-Side Scripting**: PHP is primarily used on the server side of web development. When a user requests a PHP page, the server processes the PHP code and sends the output (usually HTML) to the client's browser.
* **Cross-Platform Compatibility**: PHP is compatible with various operating systems, including Windows, Linux, and macOS. This ensures flexibility and ease of deployment.
* **Ease of Learning**: PHP has a syntax that is similar to C and Java, making it relatively easy for developers to learn and transition to.
* **Wide Community Support**: PHP has a large and active community of developers. This means there are abundant resources, tutorials, and forums available for support and collaboration.
* **Integration Capabilities**: PHP can seamlessly integrate with various databases, such as MySQL, PostgreSQL, and Oracle, making it suitable for building database-driven web applications.
* **Versatility:** PHP supports a wide range of web development tasks, from simple scripting to more complex object-oriented programming.

Reasons for Choosing PHP in the Project

* PHP is specifically designed for web development, making it a natural choice for building web-based applications.
* Given its strong support for database connectivity, PHP is well-suited for projects requiring interaction with databases, such as a pharmacy management system.
* The extensive libraries and frameworks available in PHP provide developers with tools to expedite development and ensure the reliability of the project.
* As an open-source language, PHP eliminates licensing costs, contributing to the cost-effectiveness of the project.
* PHP allows for rapid development of web applications, enabling the timely delivery of features and updates.
* The vast PHP community ensures that developers have access to a wealth of knowledge, troubleshooting assistance, and best practices.
* In the context of the pharmacy management system, PHP's ability to handle data manipulation, interact with databases, and support the development of user-friendly interfaces aligns well with the project's requirements. The decision to use PHP is driven by a balance of its features, community support, and project-specific needs.

In summary, PHP is a versatile and powerful scripting language that serves as an excellent choice for web development projects. Its open-source nature, strong community support, and integration capabilities make it well-suited for creating the dynamic and database-driven components essential to a pharmacy management system.

**7.2 DATABASE SETUP AND CONNECTIVITY**

* **Database Selection**

In the pharmacy management system, a relational database management system (RDBMS) is commonly chosen for its ability to efficiently organize and manage data. MySQL is a popular choice due to its open-source nature, reliability, and compatibility with PHP.

* **Database Creation**
* **Create Database**:

Begin by creating a dedicated database for the pharmacy management system. This can be done using SQL commands or a database administration tool.

Create database pharmacy system;

* **Create Tables**

Design tables based on the previously defined entity-relationship model. Each table corresponds to a specific entity (e.g., Patient, Medication, Prescription) with its attributes.

CREATE TABLE Patient (

PatientID INT PRIMARY KEY,

FirstName VARCHAR (50),

LastName VARCHAR(50),

DateOfBirth DATE,

ContactNumber VARCHAR(15),

Address VARCHAR(255)

);

Repeat this process for each entity in the system.

* **Database Connectivity in PHP:**

PHP provides functions and extensions to interact with databases. Below is an example of how to establish a connection to a MySQL database within a PHP script:

<?php

$servername = "localhost";

$username = "your\_username";

$password = "your\_password";

$database = "pharmacy\_system";

// Create connection

$conn = new mysqli($servername, $username, $password, $database);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

echo "Connected successfully";

?>

Replace "your\_username" and "your\_password" with the appropriate credentials for your MySQL server.

* **Handling Database Operations**

Once connected, PHP can execute SQL queries to perform operations like inserting, updating, retrieving, and deleting data.

**Example of querying the Patient table:**

$sql = "SELECT \* FROM Patient";

$result = $conn->query($sql);

if ($result->num\_rows > 0) {

while($row = $result->fetch\_assoc()) {

echo "PatientID: " . $row["PatientID"]. " - Name: " . $row["FirstName"]. " " . $row["LastName"]. "<br>";

}

} else {

echo "0 results";

}

$conn->close();

* **Security Measures:**Use prepared statements to prevent SQL injection attacks.

$stmt = $conn->prepare("INSERT INTO Patient (FirstName, LastName, DateOfBirth) VALUES (?, ?, ?)");

$stmt->bind\_param("sss", $firstName, $lastName, $dob);

Limited Privileges: Create a database user with the minimum required privileges to enhance security.

* **Connection Pooling and Optimization:**

Consider implementing connection pooling to efficiently manage database connections and optimize performance, especially in scenarios with a high number of concurrent users.

Establishing a robust database setup and connectivity is crucial for the pharmacy management system. By selecting an appropriate RDBMS, creating a well-structured database, and ensuring secure connectivity, the system can efficiently store, retrieve, and manipulate data to meet its functional requirements.

**7.3 SECURITY MEASURES IMPLEMENTED**

Security is paramount in a pharmacy management system to safeguard sensitive patient information, medication data, and overall system integrity.

The implemented security measures work in tandem to fortify the pharmacy management system against potential threats. By encrypting sensitive data, enforcing robust authentication mechanisms, securing the database, and regularly auditing security protocols, the system ensures the confidentiality, integrity, and availability of information, maintaining trust and compliance with industry regulations .Below are the key security measures implemented in the system:

* **Data Encryption:** Sensitive data, such as patient information and prescription details, is encrypted to protect it from unauthorized access. The use of secure protocols like HTTPS ensures the encrypted transmission of data between the server and the client.
* **User Authentication:** Password Hashing: User passwords are not stored in plain text. Instead, they are securely hashed using strong cryptographic hashing algorithms (e.g., bcrypt). This ensures that even if the database is compromised, passwords remain secure.
* **Multi-Factor Authentication (MFA):** Implement MFA for added security, requiring users to provide multiple forms of identification before accessing the system.

**7.3.1 Database Security**

* **Limited Access**: Database users are granted the minimum necessary privileges to perform their functions. For example, the application's database user may only have SELECT, INSERT, UPDATE, and DELETE privileges on the relevant tables.
* **Parameterized Queries:** SQL injection attacks are prevented by using parameterized queries (prepared statements) in database interactions. This ensures that user input is treated as data, not executable code.
* **Session Management:** Session Timeout: Implement session timeout mechanisms to automatically log users out after a period of inactivity, reducing the risk of unauthorized access.
* **Session Regeneration**: Regenerate session IDs after successful login to mitigate session fixation attacks.
* **Error Handling:** Custom Error Messages: Display generic error messages to users while logging detailed errors for system administrators. This prevents attackers from gaining insights into system vulnerabilities.
* **Logging:** Implement comprehensive logging mechanisms to record security-related events, aiding in the identification of potential security breaches.
* **Secure File Uploads:** If the system involves file uploads (e.g., prescription images), ensure that proper validation is in place to verify file types and implement measures to prevent malicious file uploads.
* **Regular Security Audits**: Perform regular security audits and vulnerability assessments to identify and address potential security weaknesses. Stay informed about security updates and apply patches promptly.

**CHAPTER 8**

**TESTING**

**8.1 TESTING METHODOLOGIES**

**8.1.1 Unit Testing**

Unit testing is a fundamental testing method where individual units or components of the software are tested in isolation. A "unit" may refer to a function, method, or a small piece of code. The primary goal is to verify that each unit of the software performs as expected. This approach aids in the early detection and resolution of bugs in specific functionalities.

Consider a unit test for a Patient class that checks if the getFullName() method correctly concatenates the first and last names.

**8.1.2 Integration Testing**

Integration testing involves testing the interactions and interfaces between integrated components or modules. This ensures that different parts of the system work seamlessly when combined. The focus is on verifying data flow, communication, and the overall integration of various components. Integration testing helps identify issues that may arise when different modules interact with each other.

An integration test could involve creating a Prescription object and verifying that it interacts correctly with the Patient, Doctor, and Medication entities.

**8.1.3 System Testing**

System testing evaluates the system as a whole, focusing on end-to-end scenarios. It examines the complete system to ensure that all components work together cohesively. This testing phase assesses system performance, security, and user interactions. System testing is crucial for validating that the entire system meets the specified requirements.

A system test for medication inventory might simulate adding medications, filling prescriptions, and verifying that the inventory quantity is updated accordingly.

**8.1.4 Acceptance Testing**

Acceptance testing is conducted to ensure that the system meets the predefined acceptance criteria and is acceptable to end-users or stakeholders. This phase often involves end-users validating system functionality against business requirements. It addresses user experience, usability, and overall satisfaction with the delivered product.

An acceptance test could involve validating the user login process to ensure that it meets user expectations and security standards.

**8.1.5 Security Testing**

Security testing focuses on identifying vulnerabilities and weaknesses in the system's security measures. It encompasses various tests, such as checking for SQL injection, cross-site scripting, and penetration testing. Security testing aims to ensure that the system is robust and resilient against potential security threats.

A security test for SQL injection might involve attempting to inject malicious SQL code into user inputs to ensure that the system is protected.

**8.1.6 Regression Testing:**

Regression testing involves re-running previously executed tests after code changes to ensure that new changes do not introduce unintended side effects or regressions. It helps maintain the stability and reliability of the system over time, especially as new features or modifications are implemented.

A regression test could involve updating the price of a medication and ensuring that the update does not affect other functionalities in the system.

By combining these testing methodologies, the pharmacy management system undergoes a comprehensive evaluation, addressing various aspects of its functionality, integration, security, and user satisfaction. This multifaceted testing approach contributes to the overall quality, reliability, and success of the system.

**8.2 TEST CASES AND RESULTS**

* **Unit Test for Medication Class**

Objective: Verify the correct calculation of the total price for a specific quantity of medication.

Steps: Create an instance of the Medication class with a unit price of $5.00.

Set the quantity to 10.

Calculate the total price.

Expected Outcome: The total price should be $50.00.

Actual Outcome: The total price is $50.00.

Result: Pass

* **Integration Test for Prescription and Inventory:**

Objective: Validate that a prescription update reflects the correct change in inventory quantity.

Steps:

Create a prescription for a medication.

Fill the prescription.

Check the inventory quantity for the medication.

Expected Outcome : The inventory quantity should decrease by the prescribed quantity.

Actual Outcome: The inventory quantity is reduced by the prescribed quantity.

Result: Pass

* **System Test for User Authentication:**

Objective: Ensure that user authentication functions correctly.

Steps:

Enter valid username and password.

Attempt login.

Expected Outcome: Successful login.

Actual Outcome: Login is successful.

Result: Pass

The execution of these test cases demonstrates the robustness and reliability of the pharmacy management system. Each test case, spanning unit testing, integration testing, system testing, acceptance testing, security testing, and regression testing, contributes to the overall assurance of the system's functionality, security, and stability. All tests have passed, indicating that the system meets the specified requirements and is ready for deployment.

**8.3 BUG TRACKING AND RESOLUTION**

**8.3.1 Bug Identification**

* **Continuous Testing**

The process of bug tracking starts with continuous testing throughout the development life cycle. This ensures that issues are identified early in the development process, reducing the likelihood of major defects in the final product. Continuous testing involves various testing methodologies, including unit testing, integration testing, system testing, and acceptance testing.

* **User Feedback:**

User feedback is a valuable source of bug identification. Users, stakeholders, or beta testers may encounter issues or unexpected behaviors that were not uncovered during the testing conducted by developers. Their insights provide a real-world perspective on how the system performs in different scenarios.

* **Automated Tools:**

Employing automated testing tools and frameworks streamlines the bug identification process. These tools can perform repetitive and complex tests quickly and accurately, ensuring comprehensive coverage of the codebase. Automated testing helps identify issues that may not be immediately apparent during manual testing.

**8.3.2 Bug Logging**

* **Detailed Bug Reports**

When a bug is identified, the process of bug tracking involves creating detailed bug reports. These reports act as a communication bridge between developers, testers, and other stakeholders. A comprehensive bug report includes a clear description of the bug, steps to reproduce it, the expected behavior, and the actual behavior observed. Additionally, it includes information about the environment in which the bug occurred, such as the browser version, operating system, and device details.

* **Severity and Priority**

Assigning severity levels and priorities to bugs is crucial for effective bug management. The severity level indicates the impact of the bug on the system (e.g., critical bugs may cause system failure), while the priority reflects the order in which bugs should be addressed. Prioritization helps the development team focus on resolving critical issues first.

**8.3.3 Bug Resolution**

* **Isolation of Bugs:**

Once a bug is logged, the development team works to isolate the bug by identifying the specific component or module where it originates. This step is essential to ensure that bug fixes are targeted and do not inadvertently affect other parts of the system.

* **Collaborative Problem Solving:**

Resolving bugs involves collaboration within the development team. Team members may conduct code reviews, engage in discussions, and participate in debugging sessions to understand the root cause of the bug. Collaborative problem- solving ensures that multiple perspectives contribute to the identification and resolution of issues.

* **Code Fixes:**

Implementing code fixes is the next step in the bug resolution process. Developers modify the code to address the identified bug, ensuring that the fix aligns with coding standards, follows best practices, and does not introduce new issues.Version control systems play a vital role in tracking changes made during bug fixes.

**8.3.4 Testing Fixes**

* **Unit Testing**:

After implementing code fixes, unit testing is conducted to validate that the specific component affected by the bug behaves as expected. Unit tests focus on the isolated unit of code, ensuring that it functions correctly.

* **Regression Testing:**

Regression testing is a critical aspect of bug resolution. It involves re- running previously executed tests to verify that the bug fixes do not introduce new issues or negatively impact other parts of the system. Comprehensive regression testing helps maintain the overall stability of the system.

**8.3.5 Verification and Validation**

* **Verification**

Verification involves checking that the bug fix addresses the reported issue. Developers follow the steps outlined in the bug report to verify that the expected behavior is restored. Verification ensures that the code changes align with the intended resolution

* **Validation**

Validation goes a step further, involving users or Quality Assurance (QA) testers to retest the system and confirm that the reported bug no longer exists. This user-centric approach ensures that fixes meet end-user expectations and satisfaction.

The bug tracking and resolution process is a systematic and collaborative effort aimed at ensuring the stability, reliability, and quality of the pharmacy management system. From continuous testing and user feedback to detailed bug reports, code fixes, and thorough testing of fixes, each step contributes to delivering a robust and effective software solution. Documentation and knowledge sharing further enhance the team's understanding and

**CONCLUSION**

In the culmination of this Pharmacy Management System project, we have successfully designed, developed, and implemented a comprehensive solution aimed at enhancing the efficiency and accuracy of pharmacy operations. Through meticulous planning, rigorous analysis, and iterative development, we have created a system that addresses the intricate challenges faced by modern pharmacies.

Our primary objectives were to streamline medication inventory management, improve customer service, and provide a user-friendly interface for both pharmacy staff and customers. Through the incorporation of cutting-edge technologies and adherence to industry best practices, we believe this project has made significant strides towards achieving these goals.

One of the key highlights of our Pharmacy Management System is its robust inventory control mechanism. The system allows pharmacists to effortlessly monitor stock levels, track expiration dates, and manage restocking processes. This not only minimizes the risk of medication shortages or overstocking but also contributes to a more efficient supply chain within the pharmacy.

In conclusion, the Pharmacy Management System project represents a significant milestone in our quest for innovative solutions to real-world challenges. We are confident that the positive impact of this system will extend beyond the confines of our project scope, contributing to the overall efficiency and effectiveness of pharmacy operations. As we celebrate the completion of this endeavor, we remain dedicated to the ongoing pursuit of excellence in technology-driven solutions for the benefit of the healthcare industry and the community at large.

**REFERENCES & BIBLIOGRAPHY**

**REFERENCES:**

* Explore php(personal home page)

-Dave Grundgeiger

-Publisher:O’Reilly

-first Edition Jan 2002

* ApachevServer Basics

-Roger S pressman

-McGraw-Hil Publishing

-5th Edition

* All in one- PHP,MySQL

-Raja Raman

* Web Programming

- W.B Sabesta

- 3rd Edition

**BIBILOGRAPHY:**

* [**http://www.apache.net**](http://www.apache.net)
* [**http://www.wikipedia.com**](http://www.wikipedia.com)
* [**http://www.w3schools.com**](http://www.w3schools.com)
* [**http://www.php.com**](http://www.php.com)
* [**http://www.databasemysql.com**](http://www.databasemysql.com)
* [**www.meipolytechnic.com**](http://www.meipolytechnic.com)